Available online at www.elixirpublishers.com (Elixir International Journal)

Computer Science and Engineering

Elixir Comp. Sci. & Engg. 51A (2012) 11115-11119

ISSN: 2229-712X

Design of memory controller based on AMBA AHB protocol

Jayapraveen.D and T.Geetha Priya

Veltech Multitech Dr.Rangarajan Dr,Sakunthala Engineering College, Avadi, TamilNadu, India

ARTICLE INFO

Article history:

Received: 25 January 2012;

Received in revised form:

13 October 2012;

Accepted: 27 October 2012;

Keywords

ARM;
AMBA;
Memory Controller,
AHB bus,
Arbiter,
Decoder.

ABSTRACT

The performance of a computer system is heavily dependent on the characteristics of it's interconnect architecture. A poorly designed system bus can throttle the transfer of instructions and data between memory and processor, or between peripheral devices and memory. This communication bottleneck is the focus of attention among many microprocessor and system manufacturers have adopted a number of bus standards. Hence memory access time has been a bottleneck which limits system performance. Memory controller (MC) is designed to tackle this problem. The Advanced Microcontroller Bus Architecture (AMBA) specification defines an on chip communications standard for designing high-performance embedded microcontrollers. This paper focuses on how to build an AMBA Advanced High performance Bus (AHB) based memory controller that can work efficiently in multi- master and multi- slave communication model.

Introduction

As the demand for more powerful and flexible computing devices increases, more and more System-on- Chip (SoC) is being developed. Many SoCs comprise Application Specific Integrated Circuits (ASICs) that are offered by several companies. The Advanced RISC Machines (ARM) microprocessor is very popular for SoC solutions. Today it is fair to say that the ARM Embedded Technology is universally recognized as an industry standard for ASIC design for portable applications. Creating and applying powerful, portable and at the same time re-usable intellectual Property (IP), capable of enhancing an ARM core is therefore of utmost importance to any ASIC design centre.

The Advanced Microcontroller Bus Architecture (AMBA) is an open standard, on-chip bus specification that details a strategy for the interconnection and management of functional blocks that makes up a SoC. AMBA defines a signal protocol for the connection of multiple blocks in a SoC. It facilitates the development of embedded processors (e.g., ARM microprocessors) with multiple peripherals. AMBA enhances a reusable design methodology by defining a common bus structure for SoC modules. SoCs, and in particular ARM-based SoCs, are well suited for communication applications, including cable modems, xDSL, Voice-over-IP (VoiP) and Internet appliances, handheld devices (e.g., Personal Digital Assistants), GSM and UMTS systems, digital video cameras, handsets, and so forth. SoCs can also be used by the automotive industries, e.g. for handling tasks inside a car.

With the popularization of the SoCs in the above mentioned communication and multimedia field, the high bandwidth requirement has become a bottle neck of the SoCs. Advanced High-performance Bus (AHB) is high performance system bus that used widely in industry, and SDRAM is the main memory for most of SoCs system. Thus, it is valuable to improve the memory access speed for the SoCs.

The AMBA AHB is for high-performance, high clock frequency system modules. The AHB acts as the high-performance system backbone bus. AHB supports the efficient connection of processors, on-chip memories and off-chip external memory interfaces with low-power peripheral macrocell functions. AHB is also specified to ensure ease of use in an eff1cient design flow using synthesis and automated test techniques. AHB multiple bus masters and provides high-bandwidth operation, and AMBA AHB implements the features required for high-performance, high clock frequency systems including burst transfers, split transactions, single-cycle bus master handover, single-clock edge operation, non-tristate implementation, and wider data bus con- figurations (64/128 bits).

Description of AMBA AHB bus.

An AMBA AHB design may include one or more bus masters, typically a system would contain at least the processor and the test interface . However, it would also be common for a Direct Memory Access (DMA) or Digital Signal Processor (DSP) to be included as bus masters. The external memory interface, the Advanced Peripheral Bus (APB) Bridge and any internal memory are the most common AHB slaves. Any other peripheral in the system could also be included as an AHB slave. However, low-bandwidth peripherals typically reside on the APB.

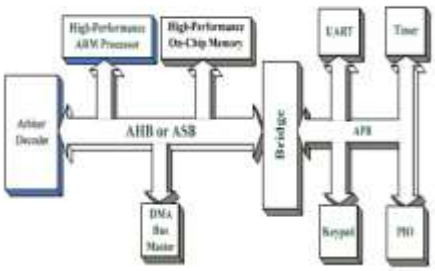


Fig. 1 A typical AMBA architecture

Tele:
E-mail addresses: jay2524@gmail.com

A typical AMBA AHB system design includes the following components:

AHB Master- A bus master is able to initiate read and write operations by providing an address and control information. Only one bus master is allowed to actively use the bus at any one time.

AHB Slave - A bus slave responds to a read or write operation within a given address-space range. The bus slave signals back to the active master the success, failure or waiting of the data transfer.

AHB Arbiter - The bus arbiter ensures that only one bus master at a time is allowed to initiate data transfers. Even though the arbitration protocol is fixed, any arbitration algorithm, such as highest priority or fair access can be implemented depending on the application requirements. An AHB would include only one arbiter, although this would be trivial in single bus master systems.

AHB decoder-The AHB decoder is used to decode the address of each transfer and provide a select signal for the slave that is involved in the transfer. A single centralized decoder is required in all AHB implementations.

The AMBA AHB bus protocol is designed to be used with a central multiplexer interconnection scheme. Using this scheme all bus masters drive out the address and control signals indicating the transfer they wish to perform and the arbiter determines which master has its address and control signals routed to all of the slaves. A central decoder is also required to control the read data and response signal multiplexer, which selects the appropriate signals from the slave that is involved in the transfer. Fig.1 illustrates the structure required to implement an AMBA AHB design with three masters and four slaves.

Before an AMBA AHB transfer can commence the bus master must be granted access to the bus. This process is started by the master asserting a request signal to the arbiter. Then the arbiter indicates when the master will be granted use of the bus. A granted bus master starts an AMBA AHB transfer by driving the address and control signals. These signals provide information on the address, direction and width of the transfer, as well as an indication if the transfer forms part of a burst. Two different forms of burst transfers are allowed: incrementing bursts, which do not wrap at address boundaries; and wrapping bursts, which wrap at particular address boundaries. A write data bus is used to move data from the master to a slave, while a read data bus is used to move data from a slave to the master.

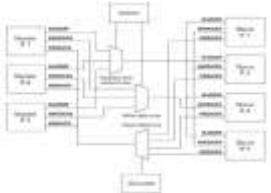


Fig.2 An AMBA AHB design with three masters and four slaves.

Every transfer consists of an address and control cycle and one or more cycles for the data. The address cannot be extended and therefore all slaves must sample the address during this time. The data, however, can be extended using the HREADY signal. When LOW this signal causes wait states to be inserted into the transfer and allows extra time for the slave to provide or sample data.

During a transfer the slave shows the status using the response signals, HRESP [l:O]:

OKAY: The OKAY response is used to indicate that the transfer is progressing normally and when HREADY goes HIGH this shows the transfer has completed success- fully.

ERROR: The ERROR response indicates that a transfer error has occurred and the transfer has been unsuccessful.

RETRY and *SPLIT*: Both the RETRY and SPLIT transfer responses indicate that the transfer cannot complete immediately, but the bus master should continue to attempt the transfer.

In normal operation a master is allowed to complete all the transfers in a particular burst before the arbiter grants another master access to the bus. However, in order to avoid excessive arbitration latencies it is possible for the arbiter to break up a burst and in such cases the master must re-arbitrate for the bus in order to complete the remaining transfers in the burst. An AHB transfer has two distinct sections: the address phase, which lasts only a single cycle; and the data phase, which may require several cycles. This is achieved using the HREADY signal. In a simple transfer with no wait states, the master drives the address and control signals onto the bus after the rising edge of HCLK and the slave then samples the address and control information on the next rising edge of the clock. After the slave has sampled the address and control it can start to drive the appropriate response and this is sampled by the bus master on the third rising edge of the clock.

This simple example demonstrates how the address and data phases of the transfer occur during different clock periods. In fact, the address phase of any transfer occurs during the data phase of the previous transfer. This overlapping of address and data is fundamental to the pipelined nature of the bus and allows for high performance operation, while still providing adequate time for a slave to provide the response to a transfer.

Every transfer can be classified into one of four different types, as indicated by the HTRANS [l :0] signals as shown in Table 1.

Furthermore, the AHB supports BURST transfer. Four, eight and sixteen-beat bursts are defined in the AMBA AHB protocol, as well as undefined-length bursts and single transfers. Both incrementing and wrapping bursts are sup- ported in the protocol. Incrementing bursts access sequential locations and the address of each transfer in the burst is just an increment of the previous address. For wrapping bursts, if the start address of the transfer is not aligned to the total number of bytes in the burst then the address of the transfers in the burst will wrap when the boundary is reached. For example, a four-beat wrapping burst of word (4-byte) accesses will wrap at 16-byte boundaries. Therefore, if the start address of the transfer is Ox34, then it has four transfers to addresses Ox34, Ox38, Ox3C and Ox30.

Burst information is provided using HBURST [2:0] and the eight possible types are defined in Table 2.

The burst size indicates the number of beats in the burst, not the number of bytes transferred. The total amount of data transferred in a burst is calculated by multiplying the number of beats by the amount of data in each beat, as indicated by HSIZE[2:0]. There are certain circumstances when a burst will not be allowed to complete and therefore it is important that any slave design which makes use of the burst information can take the correct course of action if the burst is terminated early.

Detailed description Of the Memory Controller

According to the present architecture, there is

provided a memory control apparatus including at least one memory access master for issuing a memory access instruction including a HLEN signal that represents the burst length of the transmitting data; and a memory access controller for controlling the access to the memory on the basis of the HLEN signal generated by the memory access master.

According to another aspect of the present architecture, there is provided a memory access controller having at least one memory access slave for receiving a memory access instruction issued by corresponding memory access master, generating a memory access request and feeding the information of the memory access controller back to the corresponding memory access master, the memory access instruction issued by the corresponding memory access master includes a HLEN signal that represents the burst length of the transmitting data; at least one HLEN signal decoder for decoding the HLEN signal included in the memory access instruction issued by the corresponding memory access master; an arbiter for receiving the memory access request generated by the memory access slave and sorting the received memory access requests to generate sequential access commands; a command buffer for sequentially storing the access commands generated by the arbiter; and a command controller for reading the access command stored in the command buffer and generating a memory access instruction to control the transmission of the data.

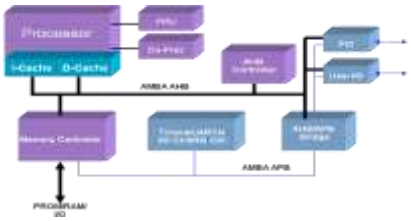


Fig.3 Structure of the SoC system with Memory controller

According to a further aspect of the present structure there is provided a memory access control method comprising the steps of issuing at least one memory access instruction including a HLEN signal that represents the burst length of the transmitting data; and controlling the access to the memory on the basis of the HLEN signal. There is also provided a memory access control method comprising receiving a memory access instruction, the memory access instruction includes a HLEN signal that represents the burst length of the transmitting data; generating a memory access request on the basis of the memory access instruction; decoding the HLEN signal; receiving the memory access request and sorting the received memory access requests to generate sequential access commands; sequentially storing the access commands; and reading the access command and generating a memory access instruction to control the transmission of the data.

Computer programs for implementing the above memory access control methods are also provided. In addition, computer program products stored on at least one computer readable medium comprising the program codes for implementing the above said memory access control methods are also provided.

The enhanced AHB according to the present structure adds one signal HLEN [3:0] from AHB masters to slave, to indicate the actual burst length of the transfer from 1 to 16. The enhanced AHB according to the present structure resolves the cycle waste issues and improves the performance simply for those transfers being not 1, 4, 8 or 16 transfer, and it is back compatible the AHB protocol and needs only very small change. The enhanced ARB according to the present structure is briefly summarized as following.

1) To give another signals HLEN [3:0], which rep- resent burst length from 1 to 16 respectively. The burst length = HLEN+l. The HLEN keeps the same cycle as HBURST. It will assert in AHB address phase by AHB master and sampled by AHB slave when HTRANS-NON- SEQ in the first data phase.

2) For fixed burst length transfer, the HLEN should be equal with the original HBURST length if the burst length transfer is unknown for the AHB master in some cases.

3) For increment unfixed burst length transfer, the HLEN will be ignored by AHB slave. It is suggested that increment usage should be avoided except the burst length is larger than 16.

4) It is option for memory controller to add HLEN_ EN, which choose whether the HLEN or HBURST will be used as burst length to back compatible AHB.

5) The HBURST will be kept to back compatible AHB, and gives information about wrap, increment and single transfer.

The AHB master portion has a plurality of AHB masters, which send out access requests to the SDRAM memory controller. The AHB interface portion comprises: a plurality of AHB slaves arranged corresponding to the plurality of AHB masters respectively, which receive the access requests from the plurality of AHB masters , and issue requests to the arbiter when the HTRANS is NONSEQ; a plurality of HLEN decoders , arranged corresponding to the plurality of AHB masters respectively, which decode the AHB control signals and the HLEN signals and send the decoded signals as well as other AHB control signals to the arbiter of the memory controller. The AHB interface portion also receives the feedback information from the controller core portion, processes the received information and sends back to the AHB master portion.

The controller core portion mainly includes: an arbiter that receives the requests from the respective AHB slaves , sorts these requests, selects and sends the AHB command to the command buffer through the command & address MUX ; a command buffer that sequentially stores the plurality of commands from the AHB interface portion; and a command controller that reads the command stored in the command buffer , generates corresponding memory access command for accessing the memory and controls the data transfer. The AHB master drives the bus address, control signals and HLEN signals at the rising edge of the clock. The respective AHB master can decide whether to issue the HLEN signals or not on the basis of its situation.

Next, the AHB slave sample the bus address, control signals and the HLEN signals at the next rising edge of the clock. If the HTRANS signal is NONSEQ, the AHB slave issues a request to the arbiter. Then, the HLEN decoder judges whether the HLEN_EN signal is 1. If the HLEN_EN is 1, the AHB slave selects the HLEN as the burst length. Otherwise, the AHB slave selects the decoded HBURST signal as the burst length. The decoder of the HBURST signal also generates the related INCR, WRAP, FULL_PAGE signal described in the above table 2, so as to indicate the type of the burst. All these signals and the other AHB control signals are send to the command & address MUX.

After that, the arbiter in the memory controller samples the

request signals of the AHB slave, sorts all the transfer requests, selects one of the requests and send the control signals related to the selected request to the command buffer in the memory controller.

Next, the command controller in the memory controller, on the basis of the current operation status of the memory and the non performed command status (for read or written command, it also includes the information on belonging to which bank and line and the information indicating the HLEN length) in the command buffer, re-sorts the commands by using optimum arithmetic and issues the next command at suitable timing to mask the waiting cycles. If the type of the current operated AHB request is INCR, the issuing of the next command is prohibited and the burst length is ignored, since it is unknown when the current command will be finished. At the same time, the memory controller also monitors the HTRANS signal of the AHB master current performing memory access. If the HTRANS signal is NONSEQ or IDLE, it indicates that the AHB asks for interrupting the current transfer, and then the memory controller issues the next command (if the next command has not been sent out).

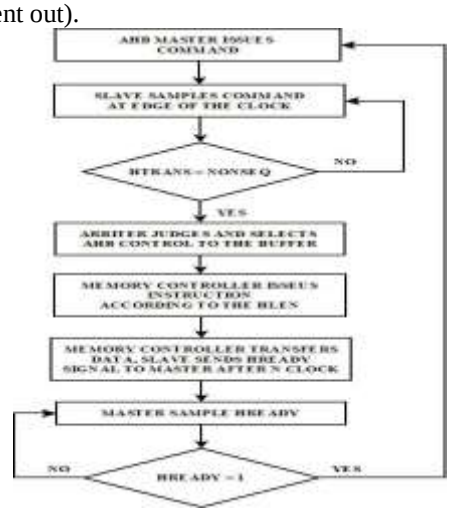
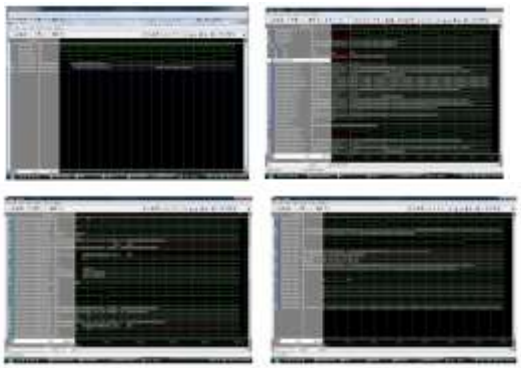


Fig.4.Flowchart of the AMBA AHB memory controller.

Then, the memory controller reads data from the memory or writes data to the memory according to the timing sequence of the memory. After the memory controller reads the data, it sends the read data to the AHB. Then, the AHB slave samples and drives the response signal to set HREADY as 1, so as to inform the AHB master that the data transfer has been finished. Next, the AHB master samples the HREADY signal. And the AHB master judges whether the HREADY signal is 1. If the HREADY signal is 1, the AHB master issues the next command and the process returns back to the initial step.

Simulation



Conclusion

In the prior AHB, only when the first data arrives, can the burst transfer completion be known, and thus the data will be received at the third cycles after issuing the command. However, according the enhanced AHB of the present structure, since the end time can be known at the first cycle of the burst, the command of the other masters can be send out in advance, and thus the data D1 can be arrived two cycles earlier than the prior AHB. Accordingly, the SDRAM access performance can be improved.

The enhanced AHB bus according to the present structure can be implemented at RTL level by modifying original AHB memory controller. SDRAM bus utilization can be increased. That means the performance can be improved. Based on different application case, a 5% to 15% performance improvement can be achieved. The simulation is built on only two AHB masters work at the same time, if more masters added; the bus utilization can be improved from 10-20% for typical multimedia application. Because most current designs are based on AHB design, the enhanced ARB protocol is very valuable because it improves the memory system performance dramatically with only very small change, and it is especially important for multimedia application when the memory access becomes the system bottle neck. It also is very convenient for AXI master to be used in an AHB bus system with such enhanced bus performance with very low performance loss compared to AXI protocols.

References

1). Building an AMBA AHB compliant Memory Controller, Hu Yueli, Yang Ben, 2011 Third International Conference on Measuring Technology and Mechatronics Automation.

2). "AMBA Specification (Rev2.0)", ARM Inc.

3). PrimeCell AHB SRAM/NOR Memory Controller, Technical Reference Manual, ARM Inc.

4). AHB Example AMBA System, Technical Reference Manual, ARM Inc.

5). Carter, J.; Hsieh,W., "Impulse: building a smarter memory controller", High-Performance Computer Architecture, Dept. Of Comput. Sci., Utah Univ., Salt Lake City, UT.

6). Clifford E. Cummings, "Synthesis and scripting Techniques for Designing Multi-Asynchronous Clock Designs", Sunburst Design, Inc.

7). Design and Analysis of Dynamically Configurable Bus Arbiters for SoCs S.Hema Chitra, P.T.Vanathi, ICGST- PDCS, Volume 8, Issue 1, December 2008.

8). Simulation and Synthesis Techniques for Asynchronous FIFO Design, Clifford E. Cummings, Sunburst Design, Inc.

9). AMBA dedicated DMA Controller with Multiple Masters using VHDL, Archana Tiwari & D.J.Dahigaonkar, International Journal of Information Technology and Knowledge Management, January-June 2011, Volume 4, No. 1, pp. 285-288.

Table 1

HTRANS[1:0]	TYPE	DESCRIPTION
00	IDLE	Indicates that no data transfer is required. The IDLE transfer type is used when a bus master is granted the bus, but does not wish to perform a data transfer. Slaves must always provide a zero wait state OKAY response to IDLE transfers and the transfer should be ignored by the slave.
01	BUSY	The BUSY transfer type allows bus masters to insert IDLE cycles in the middle of bursts of transfers. This transfer type indicates that the bus master is continuing with a burst of transfers, but the next transfer cannot take place immediately. When a master uses the BUSY transfer type the address and control signals must reflect the next transfer in the burst. The transfer should be ignored by the slave. Slaves must always provide a zero wait state OKAY response, in the same way that they respond to IDLE transfers.
10	NONSEQ	Indicates the first transfer of a burst or a single transfer. The address and control signals are unrelated to the previous transfer. Single transfers on the bus are treated as bursts of one and therefore the transfer type is NONSEQUENTIAL.
11	SEQ	The remaining transfers in a burst are SEQUENTIAL and the address is related to the previous transfer. The control information is identical to the previous transfer. The address is equal to the address of the previous transfer plus the size (in bytes). In the case of a wrapping burst the address of the transfer wraps at the address boundary equal to the size (in bytes) multiplied by the number of beats in the transfer (either 4, 8 or 16).

Table 2

HBURST[2:0]	TYPE	DESCRIPTION
000	SINGLE	single transfer
001	INCR	incrementing burst of unspecified length
010	WRAP4	4-beat wrapping burst
011	INCR4	4-beat incrementing burst
100	WRAP8	8-beat wrapping burst
101	INCR8	8-beat incrementing burst
110	WRAP16	16-beat wrapping burst
111	INCR16	16-beat incrementing burst